METHOD FOR EFFICIENT DATA ENCODING AND DECODING

This is a division of Ser. Nr. 09/967,948, Filed 2001 Oct. 2.

CROSS-REFERENCE TO RELATED APPLICATIONS

Not Applicable

STATEMENT REGARDING FED SPONSORED R & D

Not Applicable

REFERENCE TO SEQUENCE LISTING, A TABLE, OR A COMPUTER PROGRAM LISTING APPENDIX

Not Applicable

BACKGROUND OF THE INVENTION

(1) Field of Invention

This invention relates to the field of data compression and decompression. More specifically, this invention relates to data encoding and encoding.

(2) Description of Prior Art

Data compression is a process of converting data such that the converted data can be encoded in fewer bits or bytes. When being decompressed, the compressed data is recovered to its original form. Compressed data saves data storage since smaller-sized data takes less space to store. Compressed data can also enhance data transmission rates since smaller-sized data requires less time to transmit. Compression techniques have long been used in storage media such as tapes, disks, and flash cards, as well as communication equipment, such as the modem and Ethernet. With the rapidly growing demand for the Internet as well as handheld and wireless devices, compression plays an even bigger role than ever in terms of data transmission and storage.

As shown in FIG. **1**, a typical LZ data compression method processes an input data stream **10** to generate a compressed output data stream **17** by comparing an unprocessed data portion **12** of the input data stream **10** to already processed data in a history buffer **11**. If a data string **13** found in the history buffer **11** matches a current data string **14** in the unprocessed data portion **12**, the current data string **14** is replaced by a pointer (P, L) that corresponds to an offset P **15** and a match length L **16**. The offset P **15** and match

length L **16** are then encoded individually with either a fixed-length coding or a variable-length coding process. Thus, the current data string **14** is represented by a shorter coded pointer data (P, L) **18** in the compressed output data stream **17**.

As shown in FIG. **2**, fixed-length coding uses a fixed number of bits to encode a range of numbers. For example, if the number "7" were coded in a 13-bit fixed-length coding scheme, the encoded bit string would be "0000000000111". In another case, if the number "4096" were coded in a 13-bit fixed-length coding scheme, the encoded bit string would be "1000000000000". Thus, an n-bit fixed-length coding scheme encodes up to 2^n numbers ranging from 0 to $(2^n - 1)$. Variable-length coding, on the other hand, uses a variable number of bits to encode a range of numbers. There are many different variable-length coding schemes suited to different needs. FIG. **3** illustrates an example of a variable-length coding table that encodes up to 8,191 numbers ranging from 0 to 8,190. Variable-length coding typically encodes smaller numbers in fewer bits and larger numbers in more bits. For example, coding the number "4096" using the variable-length coding table shown in FIG. **3** would take 25 bits. However, coding the number "7" using the same variable-length coding table would only take 7 bits. Thus, compared to variable-length coding, fixed-length coding generally tends to be more efficient in coding larger numbers, but less efficient in coding smaller numbers.

How to encode data efficiently is essential to achieving a good compression result. Prior methods typically encode match length L **16** with variable-length coding and offset P **15** with fixed-length coding. While match length L **16** is normally a small number, offset P **15** falls within a wide range of values determined by the size of the history buffer **11**. Offset P **15** has an upper bound that increases as the size of the history buffer **11** increases. Thus, coding offset P **15** strictly with fixed-length coding as do most of the prior methods is less desirable when offset P **15** turns out to be a small number, e.g. "7". On the other hand, coding offset P **15** strictly in variable-length coding is less desirable either when offset P **15** happens to be a large number, e.g. "4096". Thus, combining different coding schemes into a single coding scheme is able to maximize the efficiencies of data encoding.

Since smaller data requires fewer bits to encode prior methods try to reduce the number of bits to encode data through methods such as replacing a matching data string

with a pointer data (P, L) as illustrated in FIG. **1**. However, it is desirable to further reduce the value of a data before encoding.

BRIEF SUMMARY OF THE INVENTION

An offset-difference coding process for improving coding efficiencies and compression performance described in accordance with the principles of this invention comprises an encoding process and a decoding process. The encoding process of offset-difference coding encodes paired input data by first determining the greater of the two input data, then calculating the difference between the two input data, replacing the larger input data with the calculated difference, and encoding said calculated difference and the smaller input data. The encoding process of offset-difference coding also generates an indicator if the larger input data that is replaced by said calculated difference is not statistically larger than the smaller input data. The decoding process of offset-difference coding decodes encoded data in an input data stream by first detecting whether an indicator exists, then decoding the encoded data and restoring the original data in response to detecting whether an indicator exists.

An alternative embodiment of an offset-difference coding process, composite offset-difference coding, described in accordance with the principles of this invention compares the calculated difference with a predetermined first threshold if the larger input data is statistically larger than the smaller input data and compares the calculated difference with a predetermined second threshold if the larger input data is not statistically larger than the smaller input data. Only if a calculated difference does not exceed the predetermined first threshold or the predetermined second threshold, then the encoding process of composite offset-difference coding replaces the larger input data with the calculated difference, and encodes the calculated difference and the smaller input data. The encoding process of composite offset-difference coding also generates an identifier if the calculated difference does not exceed the predetermined threshold as well as an indicator if the larger input data that is replaced by the calculated difference is not statistically larger than the smaller input data. The decoding process of composite offset-difference coding decodes encoded data in an input data stream by first detecting whether an identifier exists, then detecting whether an indicator exists if an identifier is detected, and decoding the encoded data in

response to detecting whether an identifier exists or whether both an identifier and an indicator exist. The offset-difference coding process and the composite offset-difference coding process minimize the number of bits required to encode a data by further reducing the value of the data, thus improve the coding efficiencies and compression performance of LZ-based compression methods without compromising the memory resource or compression/decompression speed.

BRIEF DESCRIPTION OF DRAWINGS

FIG. **1** illustrates an example of a typical compression scheme using Ziv-Lempel coding approach;

FIG. **2** illustrates an example of a 13-bit fixed-length coding scheme;

FIG. **3** illustrates an example of a variable-length coding scheme;

FIG. **4** illustrates a detailed diagram of an offset-difference coding encoding process in accordance with the principles of this invention;

FIG. **5** illustrates a detailed diagram of an offset-difference coding decoding process corresponding to the encoding process shown in FIG. **4**;

FIG. **6** illustrates an alternative embodiment of an offset-difference coding encoding process, composite offset-difference coding encoding process, in accordance with the principles of this invention;

FIG. **7** illustrates a detailed diagram of a composite offset-difference coding decoding process corresponding to the encoding process shown in FIG. **6**;

FIG. **8** illustrates an example of a format of encoded output data produced by the encoding process of offset-difference coding in accordance with the principles of this invention; and

FIG. **9** illustrates an example of a format of encoded output data produced by the encoding process of composite offset-difference coding in accordance with the principles of this invention.

DETAILED DESCRIPTION OF THE INVENTION

FIG. **4** illustrates a detailed diagram of an offset-difference coding encoding process constructed in accordance with the principles of this invention. Offset-difference coding

encoding process **400** receives paired input data in an input data stream **401** such as, but not limited to, a stream of pointer data (P, L) corresponding to offset P **15** and match length L **16** shown in FIG. **1**. The input data stream **401** could also be a NodeCopy (N, D) or a LeafCopy (D, L) in the LZ78 family of data compression methods (see Communications of the ACM, Apr. 1989, vol.32, No. 4, pp. 490-504, Edward R. Fiala and Daniel H. Greene, "Data Compression with Finite Windows"). For each of the paired data <P, L> in the input data stream **401** offset-difference coding encoding process **400** compares the two data to determine the greater of the two data in step **410**. If data P is greater than or equal to data L, offset-difference coding encoding process **400** then calculates the difference between the two data by subtracting data L from data P and replaces data P with the calculated difference (P – L) in step **420**, encodes the calculated difference (P – L) with a preselected coding scheme in step **430** and data L with variable-length coding in step **440**, generates output data beginning with the encoded data L followed by the encoded data (P – L) in step **490**, and appends the encoded data to an output data stream **402**. If, however, data L is greater than data P, offset-difference coding encoding process **400** then subtracts data P from data L and replaces data L with the calculated difference (L – P) in step **450**, encodes the calculated difference (L – P) with variable-length coding in step **460** and data P with a preselected coding scheme in step **470**, generates an indicator i with a unique pattern in step **480**, generates output data beginning with the indicator i followed by the encoded data (L – P) and the encoded data P in step **490**, and appends the encoded data to an output data stream **402**. Offset-difference coding encoding process **400** then continues encoding next paired data <P, L> in the input data stream **401**.

FIG. **8** illustrates an example of encoded output data produced by offset-difference coding encoding process **400**, wherein C_1 **830** is a codeword of encoded data (L – P) **832** and encoded data P **833** with indicator i **831**, and C_3 **850** is a codeword of encoded data L **851** and encoded data (P – L) **852** without indicator i.

If data P is equal to data L then data P is treated as the larger data by offset-difference coding encoding process **400** shown in FIG. **4**. Data P, corresponding to offset P **15** shown in FIG. **1**, normally has a wide range of values while data L, corresponding to

match length L **16** shown in FIG. **1**, normally has a fairly small value. Therefore, statistically data P is larger than data L.

Indicator i is only generated when the larger input data that is replaced by the calculated difference is not statistically larger than the smaller input data. The offset-difference coding encoding process **400** shown in FIG. **4** assumes that data P is statistically larger than data L, thus generates indicator i in step **480** when data L is determined greater than data P in step **410**.

The preselected coding scheme for encoding calculated difference (P – L) in step **430** and data P in step **470** could be either fixed-length coding or variable-length coding. Moreover, the coding scheme for encoding calculated difference (P – L) and data P does not need to be same.

FIG. **5** illustrates a detailed diagram of an offset-difference coding decoding process corresponding to the encoding process shown in FIG. **4**. Offset-difference coding decoding process **500** decodes encoded data in an input data stream **501** by first detecting whether indicator i exists in step **510**. If indicator i **831** is detected, offset-difference coding decoding process **500** then decodes encoded data (L – P) **832** with variable-length coding to recover the calculated difference (L – P) in step **520** and next encoded data P **833** with a preselected decoding scheme to recover the original data P in step **530** (see FIG. **8**), and restores the original data L by adding the decoded data (L – P) to the decoded data P in step **540**. If, however, indicator i is not detected, offset-difference coding decoding process **500** then decodes encoded data L **851** with variable-length coding to recover the original data L in step **550** and next encoded data (P – L) **852** with a preselected decoding scheme to recover the calculated difference (P – L) in step **560**, restores the original data P by adding the decoded data L to the decoded data (P – L) in step **570**. Offset-difference coding decoding process **500** then saves decoding results <P, L> in a decoding buffer in step **580** and continues decoding the next encoded data.

FIG. **6** illustrates an alternative embodiment of an offset-difference coding encoding process, composite offset-difference coding encoding process, constructed in accordance with the principles of this invention. Composite offset-difference coding encoding process **600** receives paired input data in an input data stream **601** such as, but not limited to, a stream of pointer data (P, L) corresponding to offset P **15** and match length L **16**

shown in FIG. **1**. The input data stream **601** could also be a NodeCopy (N, D) or a LeafCopy (D, L) in the LZ78 family of data compression methods (see Communications of the ACM, Apr. 1989, vol.32, No. 4, pp. 490-504, Edward R. Fiala and Daniel H. Greene, "Data Compression with Finite Windows"). For each of the paired data <P, L> in the input data stream **601** composite offset-difference coding encoding process **600** compares the two data to determine the greater of the two data in step **610**. If data P is greater than or equal to data L, composite offset-difference coding encoding process **600** then calculates the difference between the two data by subtracting data L from data P and compares the calculated difference (P – L) with a predetermined threshold T1 in step **620**. If the calculated difference (P – L) exceeds threshold T1, composite offset-difference coding encoding process **600** then encodes data P with a preselected coding scheme in step **622** and data L with variable-length coding in step **624**, generates output data beginning with the encoded data L followed by the encoded data P in step **670**, and appends the encoded data to an output data stream **602**. If, however, the calculated difference (P – L) does not exceed threshold T1, composite offset-difference coding encoding process **600** then replaces data P with the calculated difference (P – L) in step **626**, encodes the calculated difference (P – L) with a preselected coding scheme in step **628** and data L with variable-length coding in step **630**, generates an identifier I with a unique pattern in step **632**, generates output data beginning with the identifier I followed by the encoded data L and the encoded data (P – L), and appends the encoded data to an output data stream **602**.

If data P is smaller than data L, composite offset-difference coding encoding process **600** then calculates the difference between the two data by subtracting data P from data L and compares the calculated difference (L – P) with a predetermined threshold T2 in step **640**. If the calculated difference (L – P) exceeds threshold T2, composite offset-difference coding encoding process **600** then encodes data P with a preselected coding scheme in step **642** and data L with variable-length coding in step **644**, generates output data beginning with the encoded data L followed by the encoded data P in step **670**, and appends the encoded data to an output data stream **602**. If, however, the calculated difference (L – P) does not exceed threshold T2, composite offset-difference coding encoding process **600** then replaces data L with the calculated difference (L – P) in step

646, encodes the calculated difference (L – P) with variable-length coding in step **648** and data P with a preselected coding scheme in step **650**, generates an identifier I with a unique pattern in step **652** and an indicator i with a unique pattern in step **654**, generates output data beginning with the identifier I followed by the indicator i, the encoded data (L – P), and the encoded data P in step **670**, and appends the encoded data to an output data stream **602**. Composite offset-difference coding encoding process **600** then continues encoding next paired data <P, L> in the input data stream **601**.

FIG. **9** illustrates an example of encoded output data produced by composite offset-difference coding encoding process **600**, wherein C_1 **930** is a codeword of encoded data L **932** and encoded data (P – L) **933** with identifier I **931**, C_2 **940** is a codeword of encoded data (L – P) **943** and encoded data P **944** with both identifier I **941** and indicator i **942**, and C_3 **950** is a codeword of encoded data L **951** and encoded data P **952** without either identifier I or indicator i.

If data P is equal to data L then data P is treated as the larger data by composite offset-difference coding encoding process **600** shown in FIG. **6**. Data P, corresponding to offset P **15** shown in FIG. **1**, normally has a wide range of values while data L, corresponding to match length L **16** shown in FIG. **1**, normally has a fairly small value. Therefore, statistically data P is larger than data L.

Identifier I is only generated when either the calculated difference (P – L) does not exceed threshold T1 or the calculated difference (L – P) does not exceed threshold T2, and indicator i is only generated when the larger input data that is replaced by the calculated difference is not statistically larger than the smaller input data. The composite offset-difference coding encoding process **600** shown in FIG. **6** assumes that data P is statistically larger than data L, thus generates indicator i in step **654** when data L is greater than data P.

The preselected coding scheme for encoding data P in step **622**, **642**, and **650**, and the calculated difference (P – L) in step **628** could be either fixed-length coding or variable-length coding. However, the same coding scheme has to be used for encoding data P in step **622** and **642**.

Threshold T1 used in the composite offset-difference coding encoding process **600** in FIG. **6** is predetermined based on the preselected coding scheme to be used. Threshold T1

should be set at the point when one coding scheme becomes more efficient to use than the other. Take the composite offset-difference coding encoding process **600** in FIG. **6** as an example and assumes that fixed-length coding is used as a preselected coding scheme for encoding data P in step **622** and variable-length coding is used as a preselected coding scheme for encoding data (P – L) in step **628**, then threshold T1 is calculated from the following:

$$L(I) + C_v(K) < C_f(\theta)$$

where **L(I)** is the number of bits to represent identifier **I**, $C_v(K)$ is the number of bits for encoding data (P – L) in step **628** with variable-length coding, and $C_f(\theta)$ is the number of bits for encoding data P in step **622** with fixed-length coding. When the left-hand side of the equation is smaller than the right-hand side of the equation, it is more efficient to encode data (P – L) in step **628** with variable-length coding; otherwise, it is more efficient to encode data P in step **622** with fixed-length coding. Therefore, threshold T1 is set at the largest **K** that satisfies this equation.

Threshold T2 used in the composite offset-difference coding encoding process **600** in FIG. **6** is also predetermined based on the preselected coding scheme to be used. Take the composite offset-difference coding encoding process **600** in FIG. **6** as an example and assumes that fixed-length coding is used as a preselected coding scheme for encoding data P in step **642** and variable-length coding is used as a preselected coding scheme for encoding data P in step **650**, then threshold T2 is calculated with the same principle from the following:

$$L(I) + L(i) + C_v(K) < C_f(\theta)$$

where **L(I)** is the number of bits to represent identifier **I**, **L(i)** is the number of bits to represent indicator **i**, $C_v(K)$ is the number of bits for encoding data P in step **650** with variable-length coding, and $C_f(\theta)$ is the number of bits for encoding data P in step **642** with fixed-length coding. Therefore threshold T2 is set at the largest **K** that satisfies this equation.

FIG. **7** illustrates a detailed diagram of a composite offset-difference coding decoding process corresponding to the encoding process shown in FIG. **6**. Composite offset-difference coding decoding process **700** decodes encoded data in an input data stream **701** by first detecting whether identifier I exists in step **710**. If identifier I **931**, **941** is

detected, composite offset-difference coding decoding process **700** continues detecting whether indicator i exists in step **720**. If indicator i **942** is detected, composite offset-difference coding decoding process **700** then decodes the encoded data (L – P) **943** with variable-length coding to recover the calculated difference (L – P) in step **722** and next encoded data P **944** with a preselected decoding scheme to recover the original data P in step **724** (see FIG. **9**), and restores the original data L by adding the decoded data (L – P) to the decoded data P in step **726**. If, however, indicator i is not detected, composite offset-difference coding decoding process **700** then decodes the encoded data L **932** with variable-length coding to recover the original data L in step **730** and the encoded data (P – L) **933** with a preselected decoding scheme to recover the calculated difference (P – L) in step **732**, and restores the original data P by adding the decoded data L to the decoded data (P – L) in step **734**.

If identifier I is not detected, composite offset-difference coding decoding process **700** then decodes encoded data L **951** with variable-length coding to recover the original data L in step **740** and encoded data P **952** with a preselected decoding scheme to recover the original data P in step **742**. Composite offset-difference coding decoding process **700** then saves decoding results <P, L> in a decoding buffer in step **760** and continues decoding the next encoded data.

The offset-difference coding process minimizes the number of bits required to encode a data by further reducing the value of the data. Data with smaller values require fewer bits to encode, especially in variable-length coding. Thus the offset-difference coding process can improve compression performance. The composite offset-difference coding process further extends the offset-difference coding process by comparing the calculated difference with a predetermined threshold. As a result, a more efficient coding scheme can be used to further enhance compression performance.

Offset-difference coding and composite offset-difference coding processes improve coding efficiencies and compression performance without compromising memory resources and compression/decompression speed. It is envisioned that these processes are especially suitable for compression applications in the environment of Internet where the decoding speed is critical as well as for handheld and wireless devices where the memory resources are limited.

It is understood that while the invention has been particularly shown and described with reference to specific examples, the invention should not be construed to be so limited. One skilled in the art may make various modifications with the benefit of this disclosure without departing from the spirit and scope of the invention.